Info 206: Computing

Lecture 5

Graphs

September 17, 2015

Motivating questions

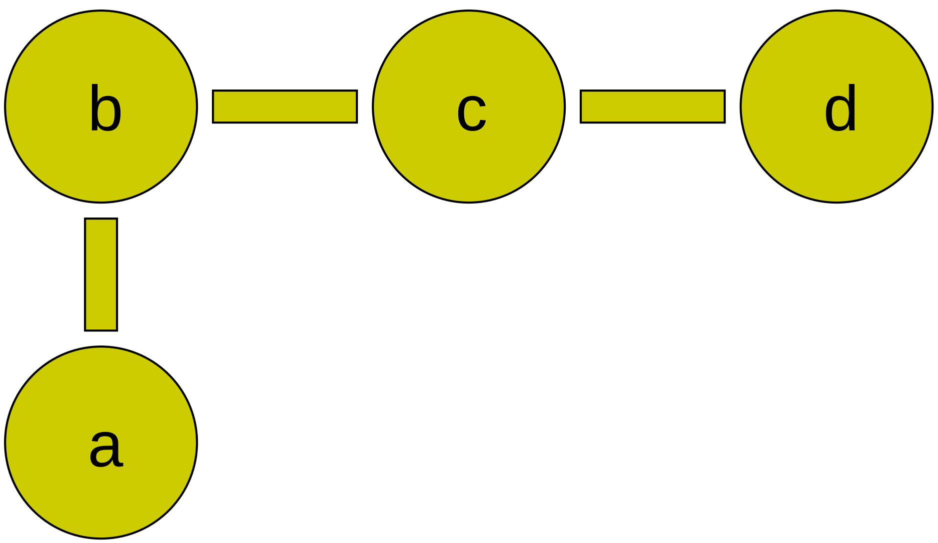
- How does Google compute page rank?
- How does the Internet decide where to move packets?

Graphs

- Graphs are more general than trees
  - Nodes or vertices
  - Edges
- Two kinds of graphs
  - Directed
  - Undirected

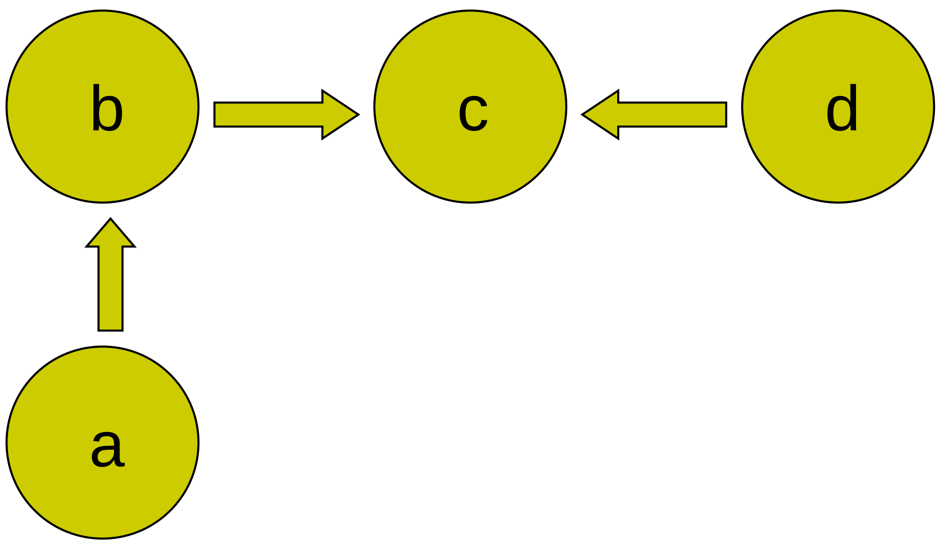
Undirected Graphs

- In an undirected graph, the edges are lines.
- Undirected graphs show a relationship between two nodes.



Directed Graphs

- In a directed graph, the edges are arrows
- Directed graphs show the flow from one node to another



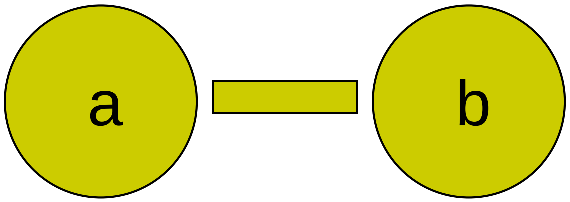
Formal Definition

- A graph $G = (V,E)$ consists of a finite set of vertices, V, and a finite set of edges E.
- Each edge is a pair (v,w) where $v, w \in V$

Directed and undirected

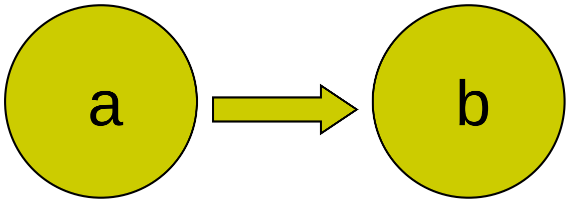
- A **directed** graph, or **digraph**, is a graph in which the edges are ordered pairs
  - (v, w) different from (w, v)
- An **undirected** graph is a graph in which the edges are unordered pairs
  - (v, w) same as (w, v)

Terminology



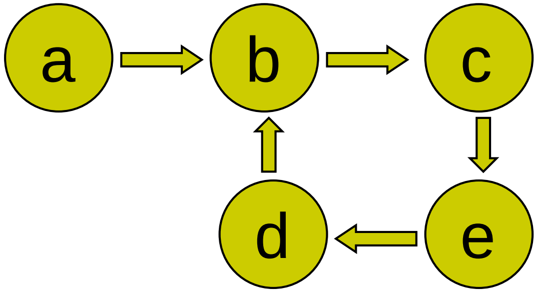
- In the undirected graph above, a and b are **adjacent** because (a,b) $\in$ E. a and b are called **neighbors.**

Terminology



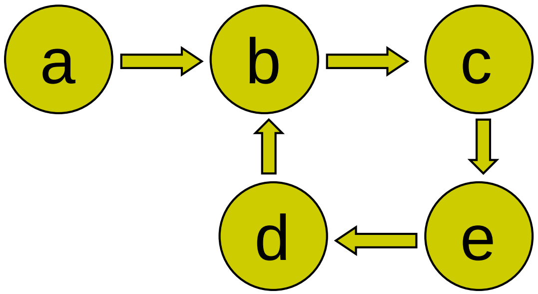
- In the directed graph above, b is **adjacent** to a because (a, b) $\in$ E. Note that a is *not* adjacent to b.
- A is a **predecessor** of node B
- B is a **successor** of node A
- The **source** of the edge is node A, the **target** is node B

Terminology



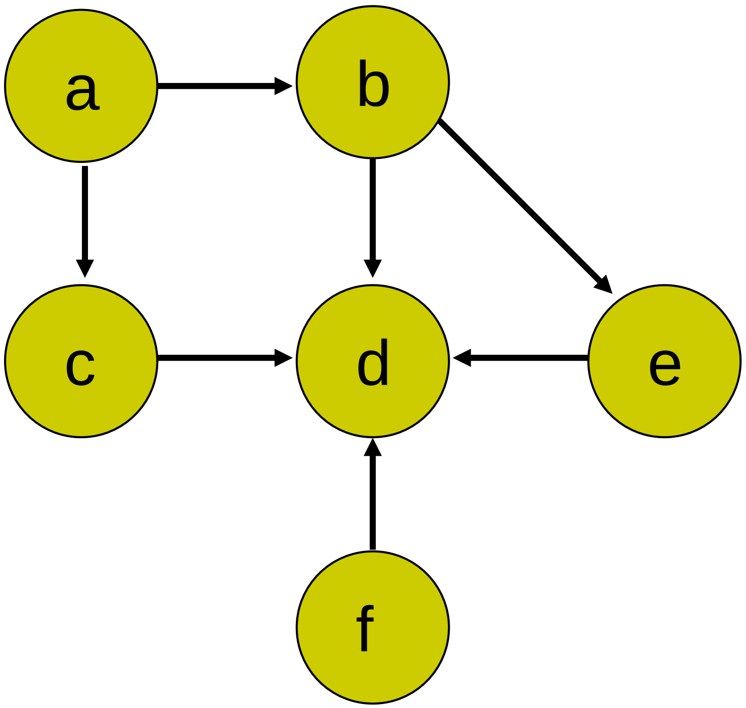
- A **path** is a sequence of vertices $w_1, w_2, \ldots w_n$ such that $(w_i, w_{i+1}) \in E$, $1 \leq i < n$, and each vertex is unique except that the path may start and end on the same vertex
- The **length** of the path is the number of edges along the path

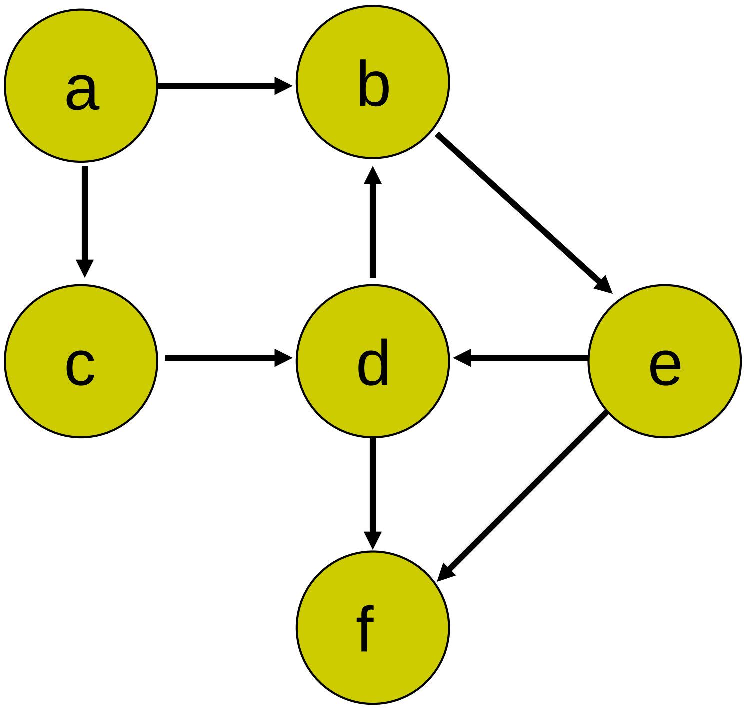
Terminology



- An **acyclic path** is a path where each vertex is unique
- A **cyclic path** is a path such that
  - There are at least two vertices on the path
  - $w_1 = w_n$ (path starts and ends at same vertex)

Cyclic or Acyclic?

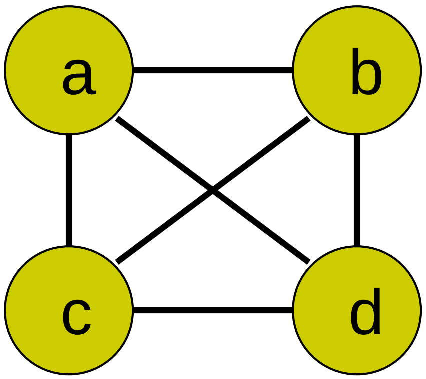




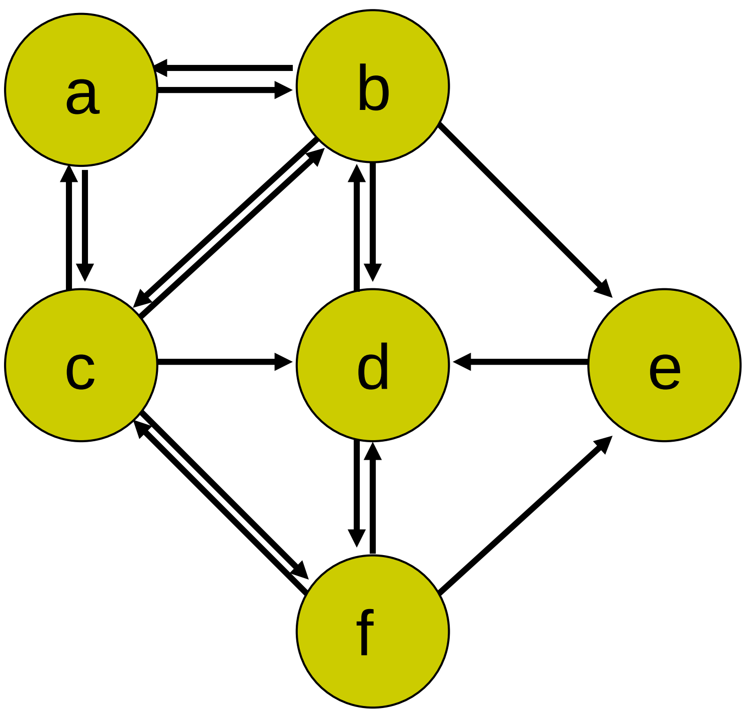
Terminology

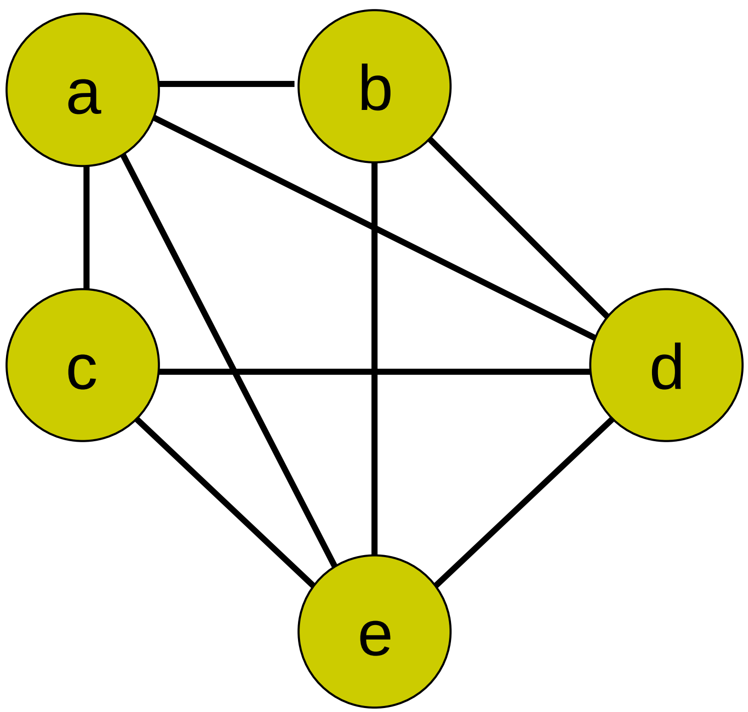
- A directed graph that has *no* cyclic paths is called a **DAG** (a Directed Acyclic Graph).
- An undirected graph that has an edge between every pair of vertices is called a **complete** graph.

Note: A directed graph can also be a complete graph; in that case, there must be an edge from every vertex to every other vertex.

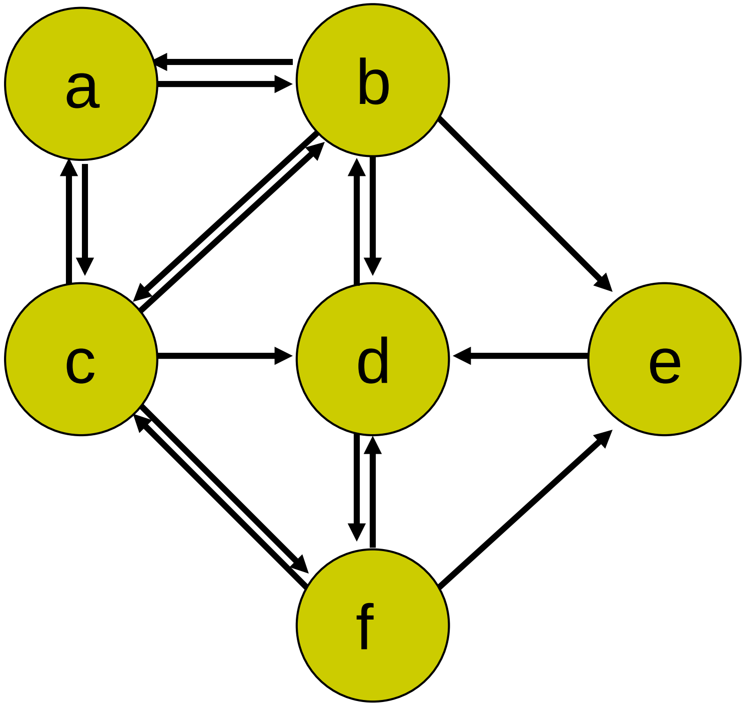


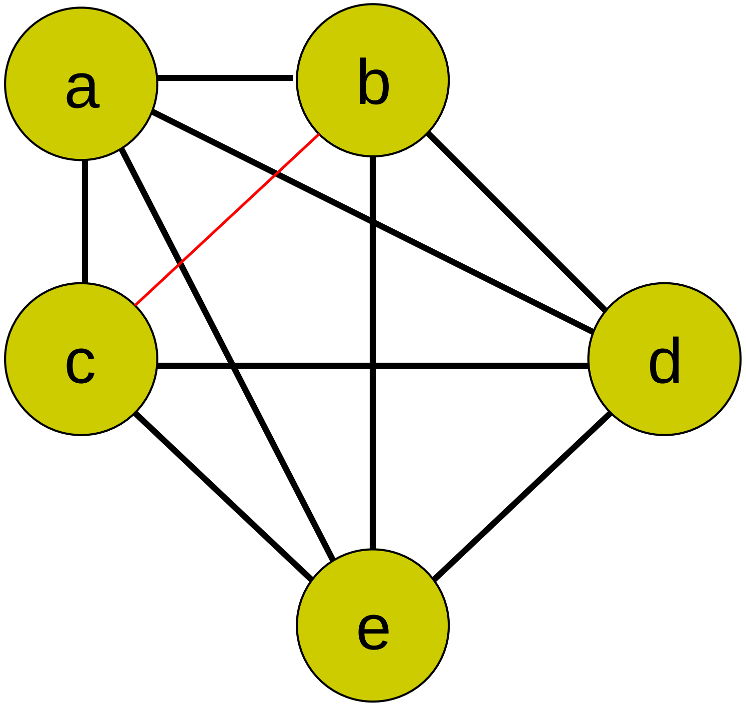
Complete or not?





Complete or not?

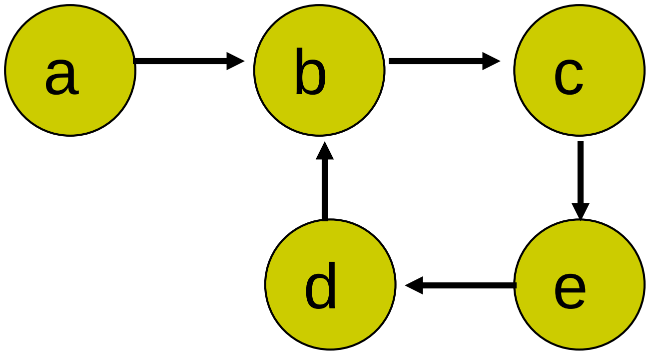


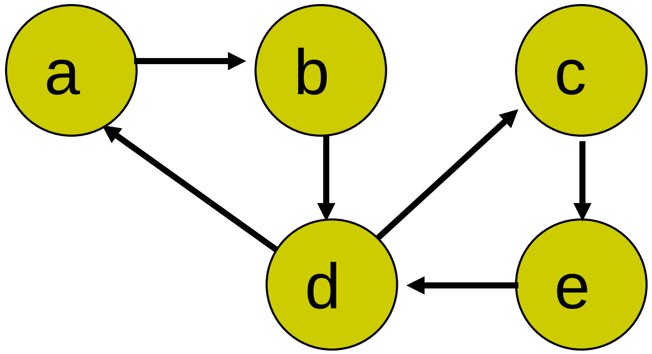


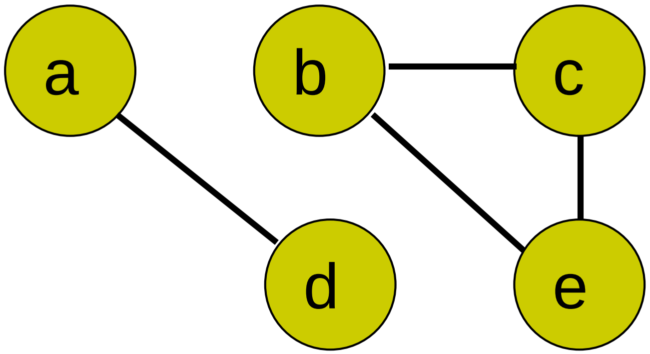
Terminology

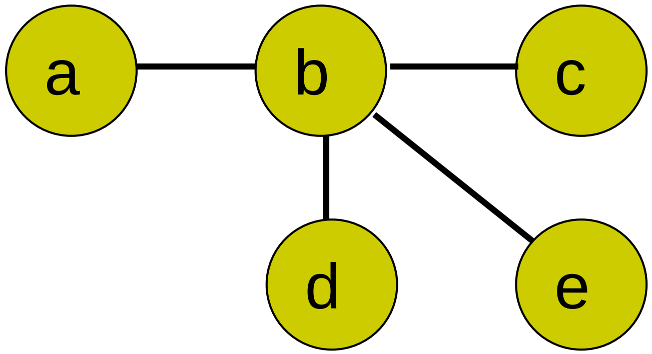
- An undirected graph is **connected** if a path exists from every vertex to every other vertex
- A directed graph is **strongly connected** if a path exists from every vertex to every other vertex
- A directed graph is **weakly connected** if a path exists from every vertex to every other vertex, disregarding the direction of the edge

Connected? (Strongly or weakly?)



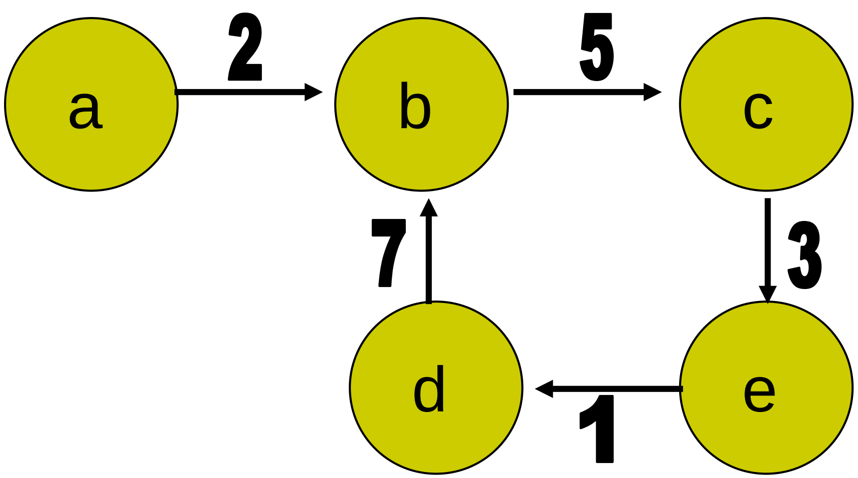






Terminology

- A graph is known as a **weighted graph** if a weight or number is associated with each edge.



Uses for Graphs

- WWW
- Computer network

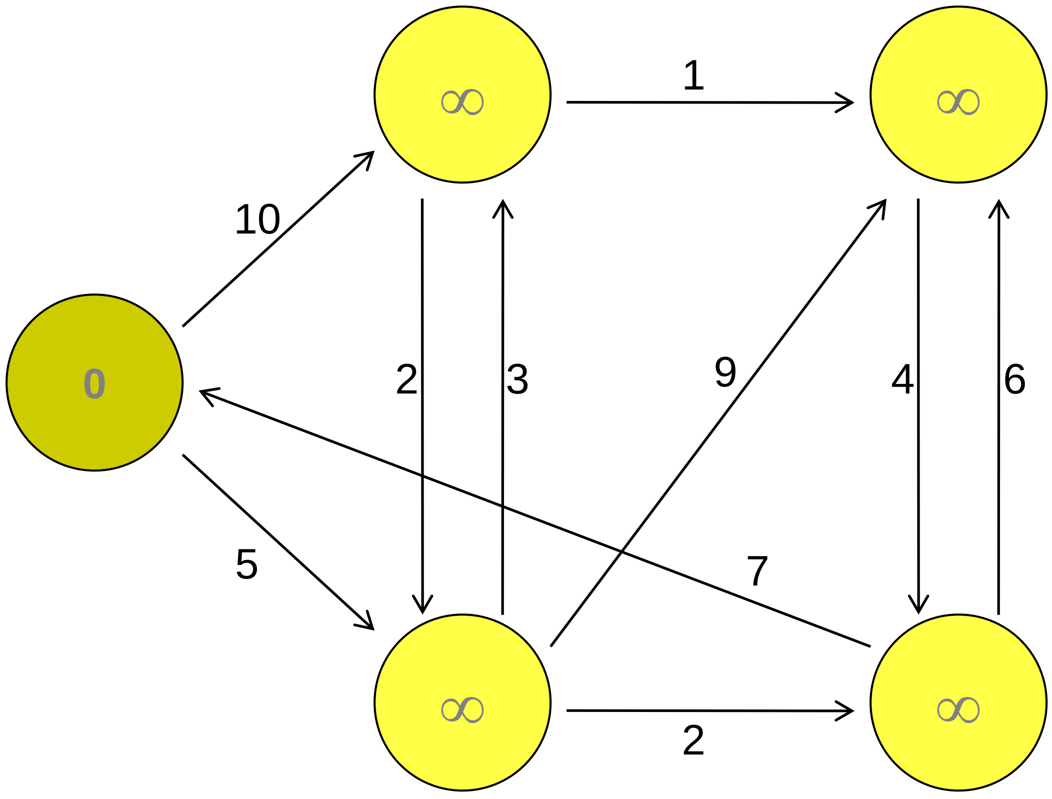
Single Source Shortest Path

- Problem: find shortest path from a source node to one or more target nodes on a weighted graph
- Dijkstra's Algorithm

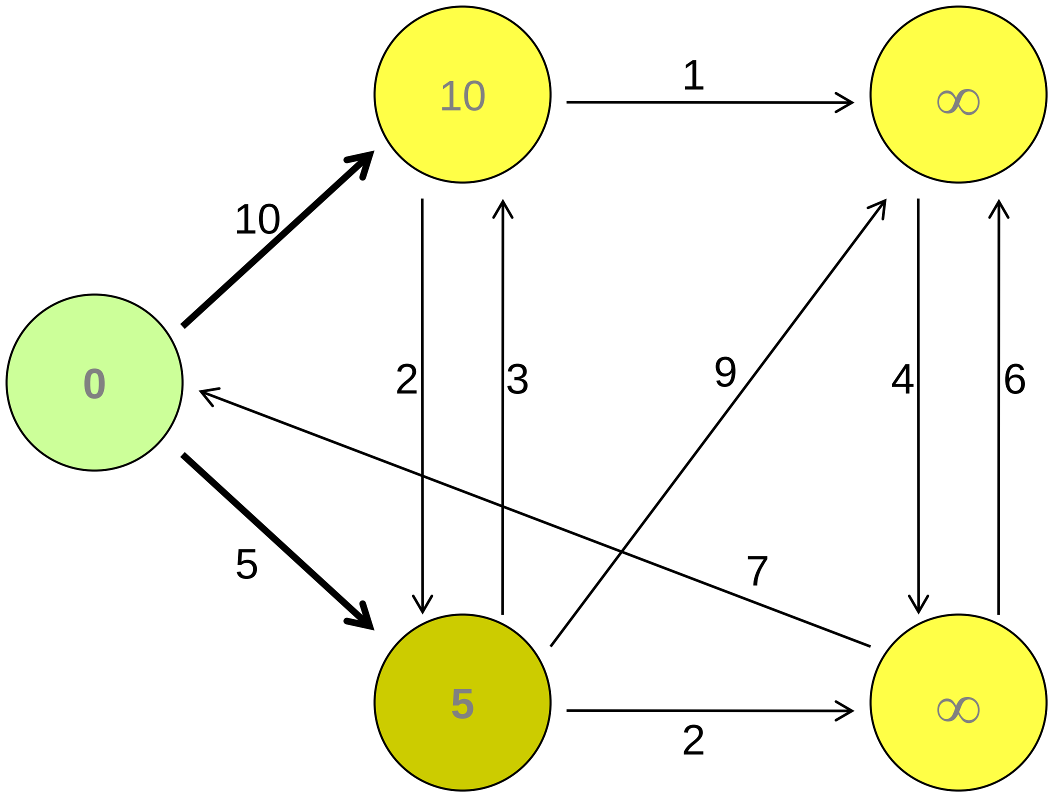
Dijsktra’s Algorithm

- Start with source vertex = 0, others = ∞
- Find smallest unvisited vertex
    - Compute new shortest distance to all unvisited nodes
    - Mark vertex as visited

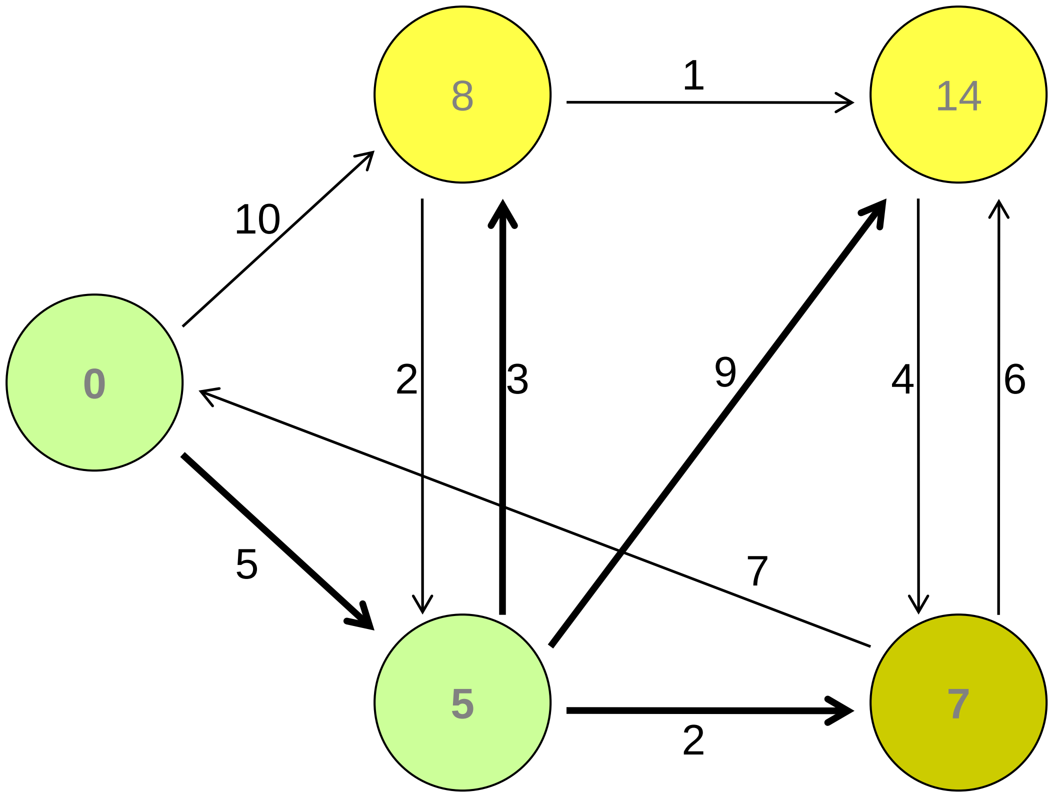
Dijkstra’s Algorithm Example



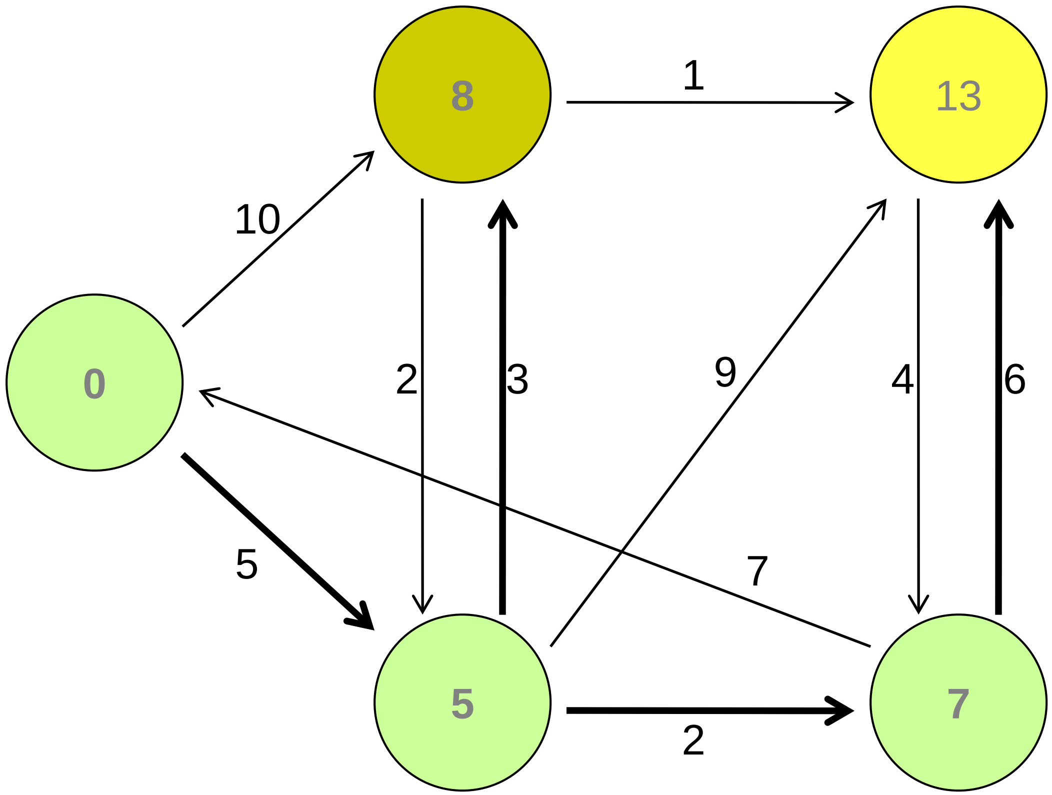
Dijkstra’s Algorithm Example



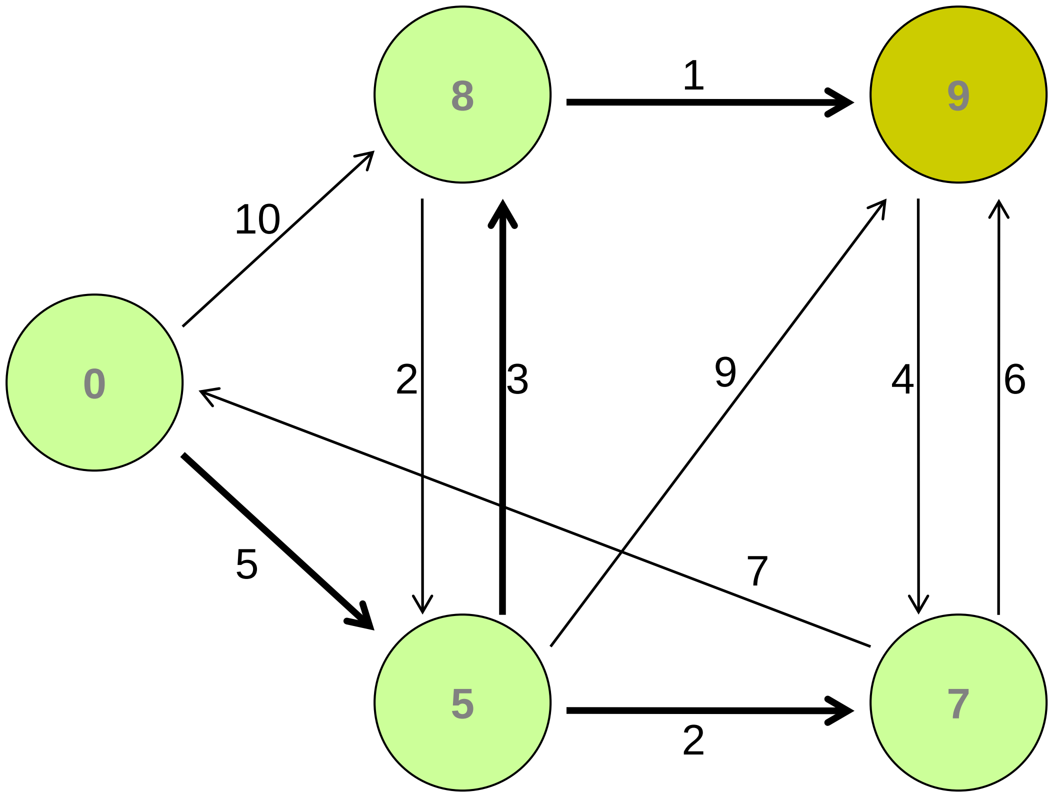
Dijkstra’s Algorithm Example



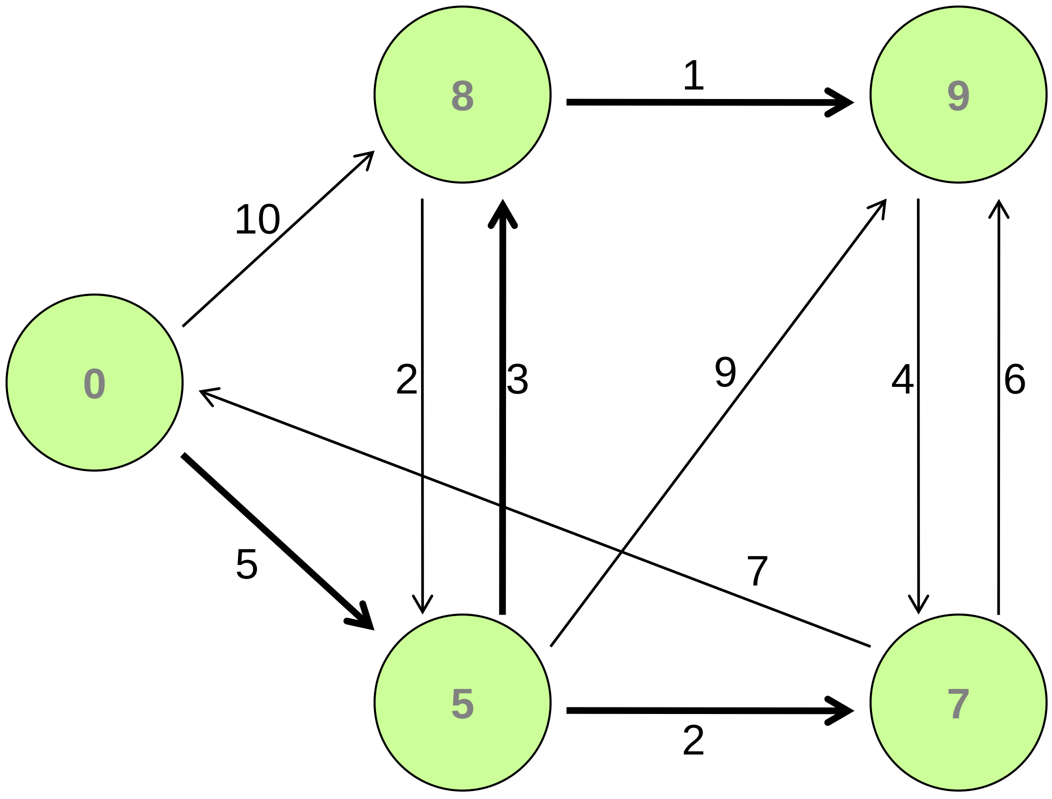
Dijkstra’s Algorithm Example



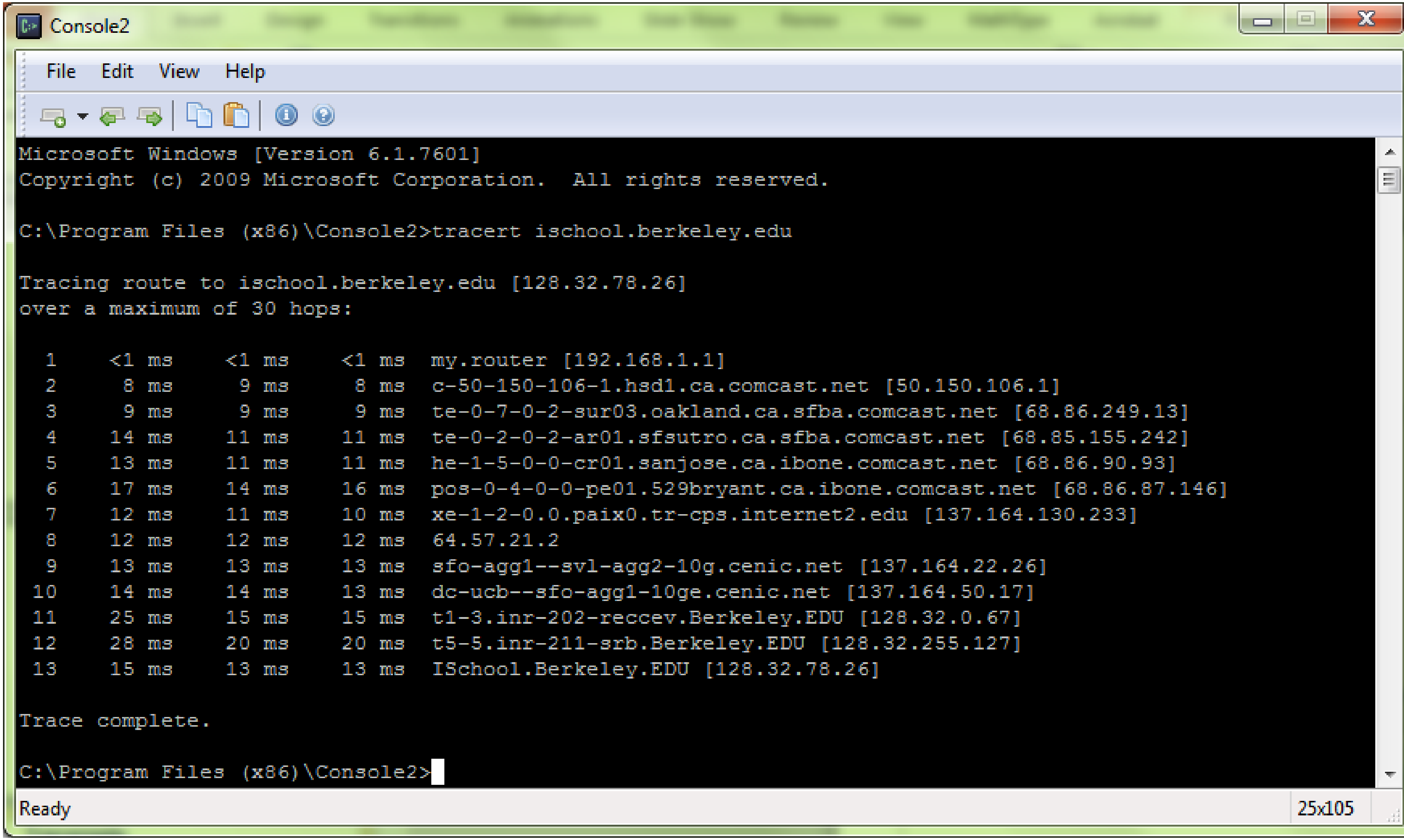
Dijkstra’s Algorithm Example



Dijkstra’s Algorithm Example

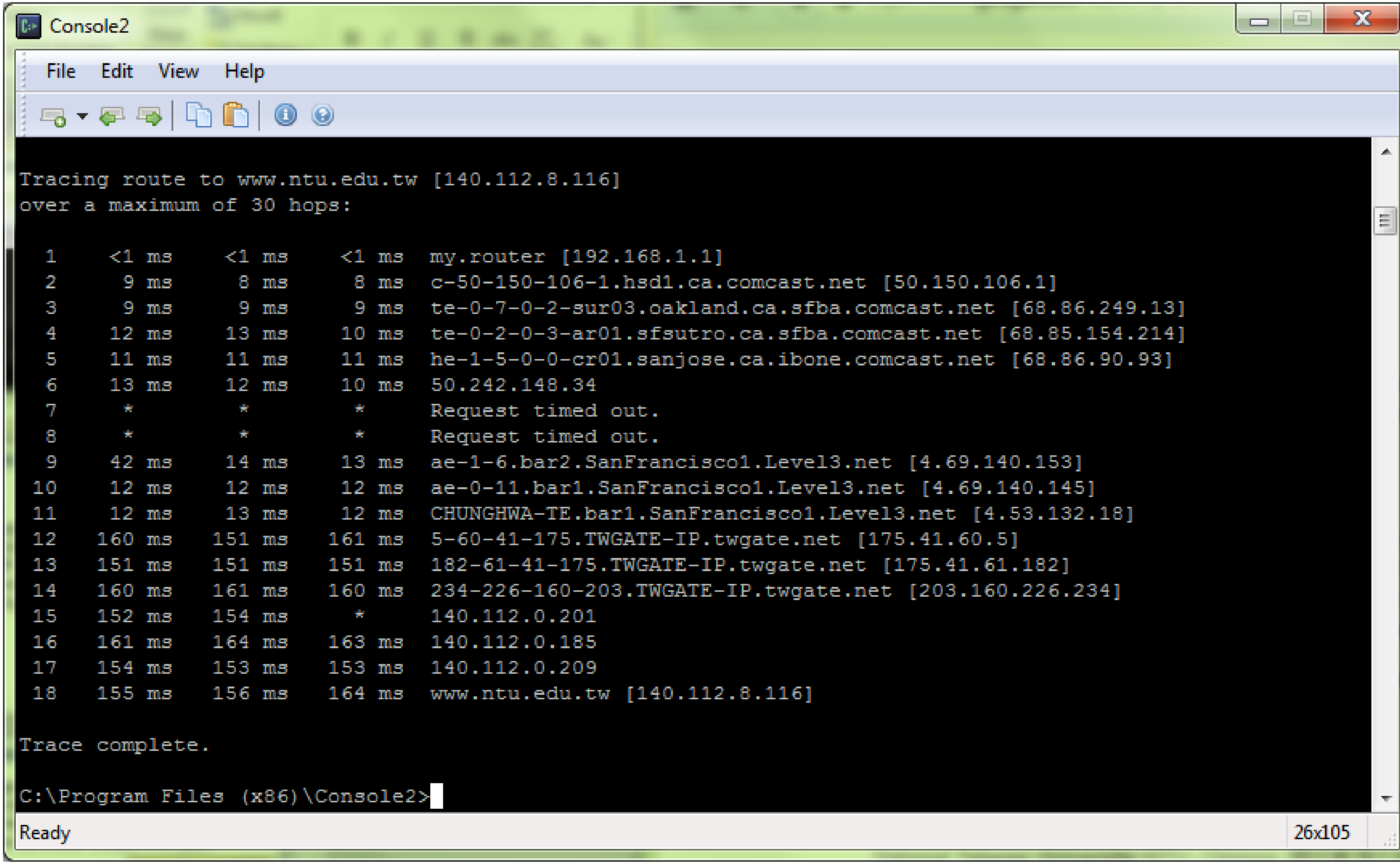


Traceroute



Console2
File Edit View Help
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Program Files (x86)\Console2>tracert ischool.berkeley.edu
Tracing route to ischool.berkeley.edu [128.32.78.26]
over a maximum of 30 hops:
  1   <1 ms   <1 ms   <1 ms my.router [192.168.1.1]
  2   8 ms   9 ms   8 ms c-50-150-106-1.hsd1.ca.comcast.net [50.150.106.1]
  3   9 ms   9 ms   9 ms te-0-7-0-2-sur03.oakland.ca.sfba.comcast.net [68.86.249.13]
  4   14 ms   11 ms   11 ms te-0-2-0-2-ar01.sfsutro.ca.sfba.comcast.net [68.85.155.242]
  5   13 ms   11 ms   11 ms he-1-5-0-0-cr01.sanjose.ca.ibone.comcast.net [68.86.90.93]
  6   17 ms   14 ms   16 ms pos-0-4-0-0-pe01.529bryant.ca.ibone.comcast.net [68.86.87.146]
  7   12 ms   11 ms   10 ms xe-1-2-0.0.paix0.tr-cps.internet2.edu [137.164.130.233]
  8   12 ms   12 ms   12 ms 64.57.21.2
  9   13 ms   13 ms   13 ms sfo-agg1--svl-agg2-10g.cenic.net [137.164.22.26]
 10   14 ms   14 ms   13 ms dc-ucb--sfo-agg1-10ge.cenic.net [137.164.50.17]
 11   25 ms   15 ms   15 ms t1-3.inr-202-reccev.Berkeley.EDU [128.32.0.67]
 12   28 ms   20 ms   20 ms t5-5.inr-211-srb.Berkeley.EDU [128.32.255.127]
 13   15 ms   13 ms   13 ms ISchool.Berkeley.EDU [128.32.78.26]
Trace complete.
C:\Program Files (x86)\Console2>
Ready
25x105

Traceroute




```
Tracing route to www.ntu.edu.tw [140.112.8.116]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  my.router [192.168.1.1]
  2     9 ms     8 ms     8 ms  c-50-150-106-1.hsd1.ca.comcast.net [50.150.106.1]
  3     9 ms     9 ms     9 ms  te-0-7-0-2-sur03.oakland.ca.sfba.comcast.net [68.86.249.13]
  4    12 ms    13 ms    10 ms  te-0-2-0-3-ar01.sfsutro.ca.sfba.comcast.net [68.85.154.214]
  5    11 ms    11 ms    11 ms  he-1-5-0-0-cr01.sanjose.ca.ibone.comcast.net [68.86.90.93]
  6    13 ms    12 ms    10 ms  50.242.148.34
  7     *        *        *     Request timed out.
  8     *        *        *     Request timed out.
  9    42 ms    14 ms    13 ms  ae-1-6.bar2.SanFrancisco1.Level3.net [4.69.140.153]
 10    12 ms    12 ms    12 ms  ae-0-11.bar1.SanFrancisco1.Level3.net [4.69.140.145]
 11    12 ms    13 ms    12 ms  CHUNGHWA-TE.bar1.SanFrancisco1.Level3.net [4.53.132.18]
 12   160 ms   151 ms   161 ms  5-60-41-175.TWGATE-IP.twgate.net [175.41.60.5]
 13   151 ms   151 ms   151 ms  182-61-41-175.TWGATE-IP.twgate.net [175.41.61.182]
 14   160 ms   161 ms   160 ms  234-226-160-203.TWGATE-IP.twgate.net [203.160.226.234]
 15   152 ms   154 ms     *     140.112.0.201
 16   161 ms   164 ms   163 ms  140.112.0.185
 17   154 ms   153 ms   153 ms  140.112.0.209
 18   155 ms   156 ms   164 ms  www.ntu.edu.tw [140.112.8.116]

Trace complete.

C:\Program Files (x86)\Console2>
```

Random Walks Over the Web

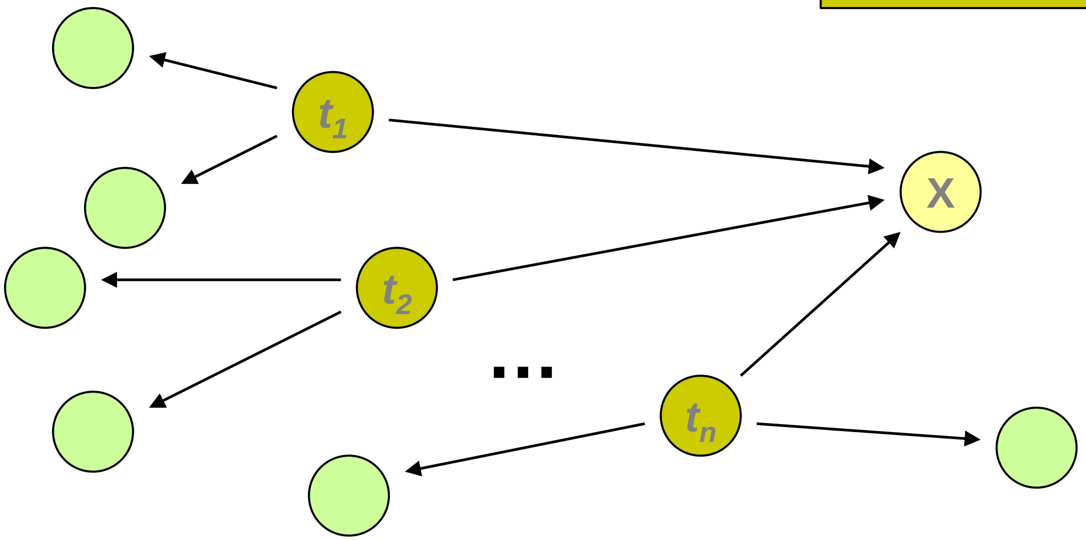
- Model:
  - User starts at a random Web page
  - User randomly clicks on links, surfing from page to page
- How much time is spent on each page?
- This is PageRank (named after Larry Page)

PageRank: Defined

Given page x with in-bound links $t_1 \ldots t_n$, where

- $C(t)$ is the out-degree of t
- α is probability of random jump
- N is the total number of nodes in the graph

$$PR(x) = \alpha\left(\frac{1}{N}\right) + (1-\alpha)\sum_{i=1}^{n}\frac{PR(t_i)}{C(t_i)}$$



PageRank

Page rank of x

Weighted probability of clicking into x from an adjacent page

$$PR(x) = \alpha\left(\frac{1}{N}\right) + (1-\alpha)\sum_{i=1}^{n}\frac{PR(t_i)}{C(t_i)}$$

Probability of a random "jumping" to x

Computing PageRank

- Properties of PageRank
  - Can be computed iteratively
  - Effects at each iteration is local
- Sketch of algorithm:
  - Start with seed PR_i values
  - Each page distributes PR_i "credit" to all pages it links to
  - Each target page adds up "credit" from multiple in-bound links to compute PR_{i+1}
  - Iterate until values converge